COFFEE SHOP SALES SQL QUERIES

KPI'S Requirements:

1. **Total Sales Analysis:**
   - **Calculate the total sales for each respective month.**

```
SELECT ROUND(sum(unit_price * transaction_qty) ) as Total_Sales FROM
coffee_shop_sales
WHERE MONTH(transaction_date) = 4;      -- (4 Means april month)
```

   - **Determine the month-on-month increase or decrease in sales.**
   - **Calculate the difference in sales between the selected month and the previous month.**

```
WITH MONTHLY_SALES AS (
   SELECT
      MONTH(transaction_date) AS Month,
      SUM(unit_price * transaction_qty) AS Total_Sales
   FROM coffee_shop_sales
   WHERE MONTH(transaction_time) IN (4, 5) -- (4 = April, 5 = May)
   GROUP BY MONTH(transaction_time)
)
SELECT
   Month,
   Total_Sales,
   ROUND(Total_Sales - LAG(Total_Sales) OVER(ORDER BY Month)) AS
Sales_Difference,
   ROUND(((Total_Sales - LAG(Total_Sales) OVER(ORDER BY Month)) /
        LAG(Total_Sales) OVER(ORDER BY Month)) * 100, 2) AS
MOM_Increase_Percentage
FROM MONTHLY_SALES
ORDER BY Month;
```

2. **Total Order Analysis:**
   - **Calculate the total number of orders for each respective month.**

```
SELECT COUNT(transaction_id) As Total_Orders
from coffee_shop_sales
WHERE Month(transaction_date)= 3 ;    -- (here 3 means march month)
```

- **Determine the month-on-month increase or decrease in the number of orders.**
- **Calculate the difference in the number of orders between the selected month and the previous month.**

```
WITH MONTHLY_ORDERS AS (
   SELECT
      MONTH(transaction_date) AS Month,
      COUNT(transaction_id) AS Total_Orders
   FROM coffee_shop_sales
   WHERE MONTH(transaction_date) IN (4, 5)
   GROUP BY Month
)
SELECT
   Month,
   Total_Orders,
   Total_Orders - LAG(Total_Orders) OVER(ORDER BY Month) AS
Order_Differences,
   ROUND(
      ((Total_Orders - LAG(Total_Orders) OVER(ORDER BY Month)) /
        LAG(Total_Orders) OVER(ORDER BY Month)) * 100, 2) AS
MOM_Percentage
FROM MONTHLY_ORDERS
ORDER BY Month;
```

3. **Total Quantity Sold Analysis:**
   - Calculate the total quantity sold for each respective month.

```
SELECT sum(transaction_qty) as Total_quantity_sold
FROM coffee_shop_sales
WHERE month(transaction_date)= 6;   -- (6 means June)
```

- Determine the month-on-month increase or decrease in the total quantity sold.

- Calculate the difference in the total quantity sold between the selected month and   the previous month.

```
   WITH MONTHLY_QUANTITY AS (
   SELECT
      MONTH(transaction_date) AS Month,
      SUM(transaction_qty) AS Total_Quantity_Sold
   FROM coffee_shop_sales
   WHERE MONTH(transaction_date) IN (4, 5)
   GROUP BY Month
)
SELECT
   Month,
    Total_Quantity_Sold,
    Total_Quantity_Sold - LAG(Total_Quantity_Sold) OVER(ORDER BY Month) AS
Total_Quantity_Difference,
    ROUND(
        ((Total_Quantity_Sold - LAG(Total_Quantity_Sold) OVER(ORDER BY Month)) /
          LAG(Total_Quantity_Sold) OVER(ORDER BY Month)) * 100,  2 ) AS
MOM_Percentage
FROM MONTHLY_QUANTITY
ORDER BY Month;
```

CHARTS Requirements:

1. **Calendar Heat Map:**
   - Implement a calendar heat map that dynamically adjusts based on the selected month from a slicer.
   - Each day on the calendar will be color-coded to represent sales volume, with darker shades indicating higher sales.
   - Implement tooltips to display detailed metrics (Sales, Orders, Quantity) when hovering over a specific day.

```
                              SELECT

                 SUM(unit_price * transaction_qty) AS Total_Sales,
                        COUNT(transaction_id) AS Total_Orders,
                       SUM(transaction_qty) AS Total_Quantity_Sold
                       FROM coffee_shop_sales
                       WHERE transaction_date;
```

EX-1 : ---
-- I want to know total_sales, total_orders, and total_quantity_orders on 27/03/2023 day?

```
select concat(round(sum(unit_price * transaction_qty)/1000,1),'k')as Total_sales,
concat(round(count(transaction_id)/1000,1),'k') as Total_order,
concat(round(sum(transaction_qty)/1000,1),'k') as Total_quantity_sold
from coffee_shop_sales
where transaction_date = '2023-03-27';
```

2. **Sales Analysis by Weekdays and Weekends:**

- Segment sales data into weekdays and weekends to analyze performance variations.
- Provide insights into whether sales patterns differ significantly between weekdays and weekends.

(weekdays= mon to fri),(weekends=sat to sun). (sun=1, mon=2, . . . sat=7)

```
SELECT

CASE

WHEN DAYOFWEEK(transaction_date) IN (1, 7) THEN 'Weekends'

ELSE 'Weekdays'

END AS Day_Type,

SUM(unit_price * transaction_qty) AS Total_Sales

FROM coffee_shop_sales

WHERE MONTH(transaction_date) = 2      -- (2 = February)

GROUP BY

CASE

WHEN DAYOFWEEK(transaction_date) IN (1, 7) THEN 'Weekends'

ELSE 'Weekdays'

END;
```

3. Sales Analysis by Store Location:

- Visualize sales data by different store locations.

- Include month-over-month (MoM) difference metrics based on the selected month in the slicer.

- Highlight MoM sales increase or decrease for each store location to ident.

```
SELECT
    store_location,
    SUM(unit_price * transaction_qty) AS Total_Sales
FROM coffee_shop_sales
WHERE MONTH(transaction_date) = 5  --( 5 = May)
GROUP BY store_location
ORDER BY Total_Sales DESC;
```

4. Daily Sales Analysis with Average Line:

- Display daily sales for the selected month with a line chart.
- Incorporate an average line on the chart to represent the average daily sales.
- Highlight bars exceeding or falling below the average sales to identify exceptional sales days.

```
WITH daily_sales AS (
    SELECT
        DATE(transaction_date) AS sales_date,
        SUM(unit_price * transaction_qty) AS total_sales
    FROM coffee_shop_sales
    WHERE MONTH(transaction_date) = 3  -- (3 = March)
    GROUP BY sales_date
),
overall_avg AS (
    SELECT
        AVG(total_sales) AS avg_daily_sales
    FROM daily_sales
)
SELECT
    daily_sales.sales_date,
    daily_sales.total_sales,
    overall_avg.avg_daily_sales,
    CASE
        WHEN daily_sales.total_sales > overall_avg.avg_daily_sales THEN
'ABOVE_AVG'
        WHEN daily_sales.total_sales < overall_avg.avg_daily_sales THEN
'BELOW_AVG'
        ELSE 'EQUAL_TO_AVG'
    END AS sales_status
FROM daily_sales
CROSS JOIN overall_avg
ORDER BY daily_sales.sales_date;
```

5. Sales Analysis by Product Category:

- Analyze sales performance across different product categories.
- Provide insights into which product categories contribute the most to overall sales.

```
SELECT
    product_category,
    CONCAT(ROUND(SUM(unit_price * transaction_qty) / 1000, 1), 'K') AS Total_Sales
FROM coffee_shop_sales
WHERE MONTH(transaction_date) = 3  -- (3 = March)
GROUP BY product_category
ORDER BY Total_Sales DESC;
```

6. Top 10 Products by Sales:

- Identify and display the top 10 products based on sales volume.

- Allow users to quickly visualize the best-performing products in terms of sales.

```
SELECT
    product_type,
    CONCAT(ROUND(SUM(unit_price * transaction_qty) / 1000, 1), 'K') AS Total_Sales
FROM coffee_shop_sales
WHERE MONTH(transaction_date) = 3  -- (3 = March)
GROUP BY product_type
ORDER BY SUM Total_Sales DESC
LIMIT 10;
```

7. Sales Analysis by Days and Hours:

- Utilize a heat map to visualize sales patterns by days and hours.

- Implement tooltips to display detailed metrics (Sales, Orders, Quantity) when hovering over a specific day-hour.

Specific day & hour in May :

```
SELECT
    MONTH(transaction_date) AS Month,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales,
    SUM(transaction_qty) AS Total_Quantity,
    COUNT(transaction_id) AS Total_Orders
FROM coffee_shop_sales
WHERE
    DAYOFWEEK(transaction_date) = 3   -- (3 = Tuesday)
   AND HOUR(transaction_time) = 8     -- (8 AM)
   AND MONTH(transaction_date) = 5   -- (5 = May)
GROUP BY Month;
```

-- Get total sales from Monday to Sunday for the month of May:

--

```
SELECT
    CASE
```

```
        WHEN DAYOFWEEK(transaction_date) = 2 THEN 'Monday'
        WHEN DAYOFWEEK(transaction_date) = 3 THEN 'Tuesday'
        WHEN DAYOFWEEK(transaction_date) = 4 THEN 'Wednesday'
        WHEN DAYOFWEEK(transaction_date) = 5 THEN 'Thursday'
        WHEN DAYOFWEEK(transaction_date) = 6 THEN 'Friday'
        WHEN DAYOFWEEK(transaction_date) = 7 THEN 'Saturday'
        ELSE 'Sunday'
    END AS Day_Of_Week,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_shop_sales
WHERE MONTH(transaction_date) = 5  --( 5 = May)
GROUP BY
    CASE
        WHEN DAYOFWEEK(transaction_date) = 2 THEN 'Monday'
        WHEN DAYOFWEEK(transaction_date) = 3 THEN 'Tuesday'
        WHEN DAYOFWEEK(transaction_date) = 4 THEN 'Wednesday'
        WHEN DAYOFWEEK(transaction_date) = 5 THEN 'Thursday'
        WHEN DAYOFWEEK(transaction_date) = 6 THEN 'Friday'
        WHEN DAYOFWEEK(transaction_date) = 7 THEN 'Saturday'
        ELSE 'Sunday'
    END
ORDER BY
    FIELD(Day_Of_Week, 'Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday');
```

-- Sales by hour (all days in May):

```
SELECT
    HOUR(transaction_time) AS Hour_Of_Day,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_shop_sales
```

```
WHERE MONTH(transaction_date) = 5  -- (5 = May)
GROUP BY HOUR(transaction_time)
ORDER BY HOUR(transaction_time);
```